**DATABASE MANAGEMENT**

**MIST.6030 (SECTION 201), FALL 2019**

**FINAL TEAM PROJECT**

Members:

Modupe Ajala

Kellye Houdagba

Company: Aras Corporation

Liaison: Jillian Jakubowicz

Contact Info: jjakubowicz@aras.com

**Business Background**

Aras Corporation is a PLM Company headquartered in Andover,Ma. As PLM stands for Product Lifecycle Management, the company sells its software, Innovator, as a service to large and medium sized corporations who use the software as a method of managing one or many products and its data over the life of the product from inception to decommission in an effort to make better business decisions. They have offices all over the world including Japan, United Kingdom, Germany, France, and Italy. These offices together have about 500 employees. Some of the companies that use Aras’ software are GE Aviation, General Motors, BAE Systems, Microsoft, and Honda. Some of Aras’ competitors are Siemens, PTC, Dassault Systems, and Oracle.

**Database Purpose**

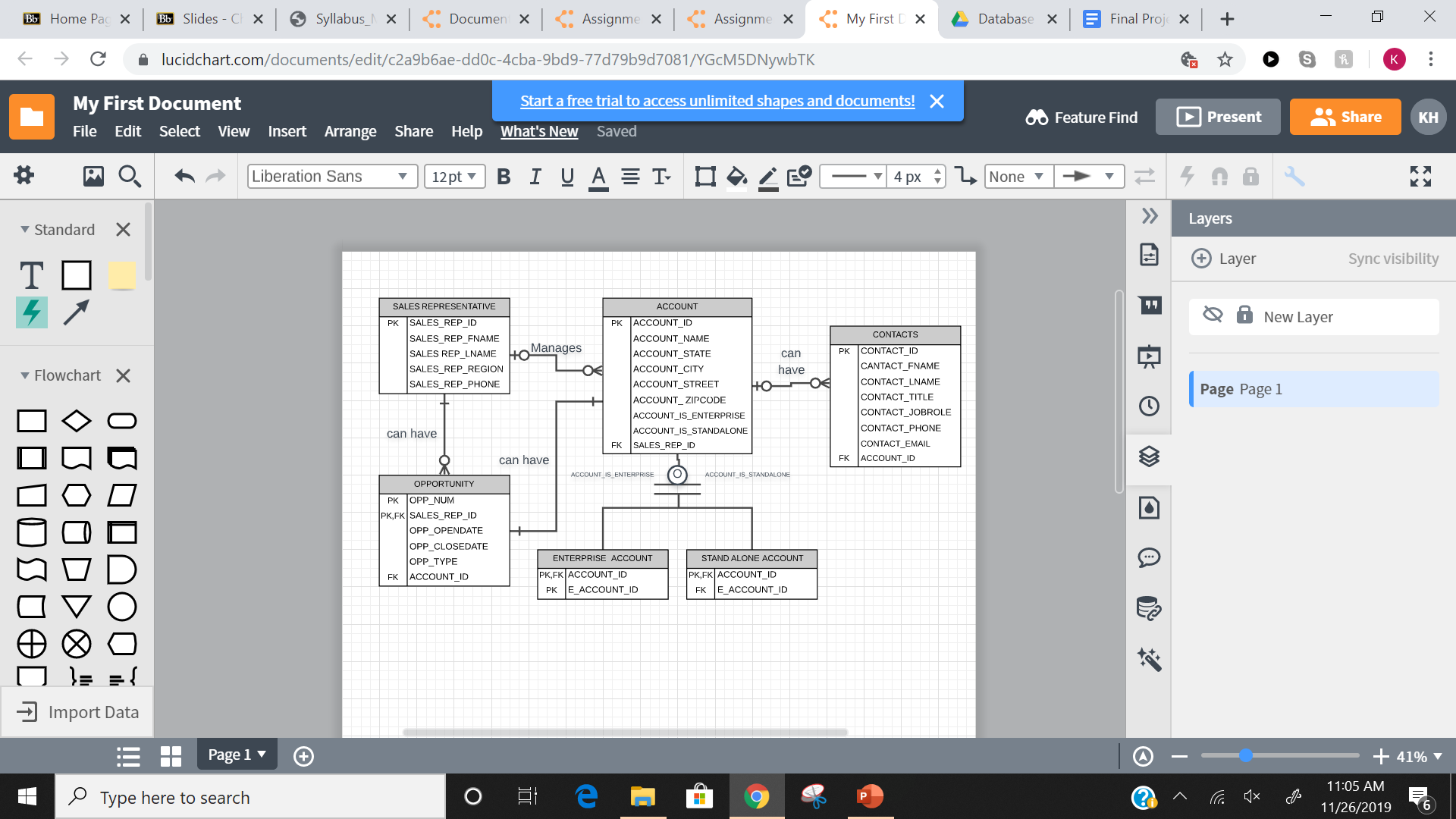
As we met with Jillian a business analyst at Aras, we discussed the objective of this project and how we could help Aras. We struggled at first to find problems that we could help solve because the company uses their own software, Innovator, as a database and they tailor it to their needs. The software is so easy to use that you can upload and change data instantly, add attributes and rules in an instant as well. So there almost wasn’t much we could do for Aras. But in our discussions we came across a problem we could help with.

Aras’ customers are very big companies like Microsoft and GE Aviation. Each of these large corporations have slightly smaller companies within them. At this moment Aras’ database is not able to view companies within companies. So our task was to find a way to roll up all of the companies that belong to a bigger company under that bigger company. Additionally, employees need to be able to view contacts within each company, see the sales rep assigned to the companies, and sales opportunities associated with each account.

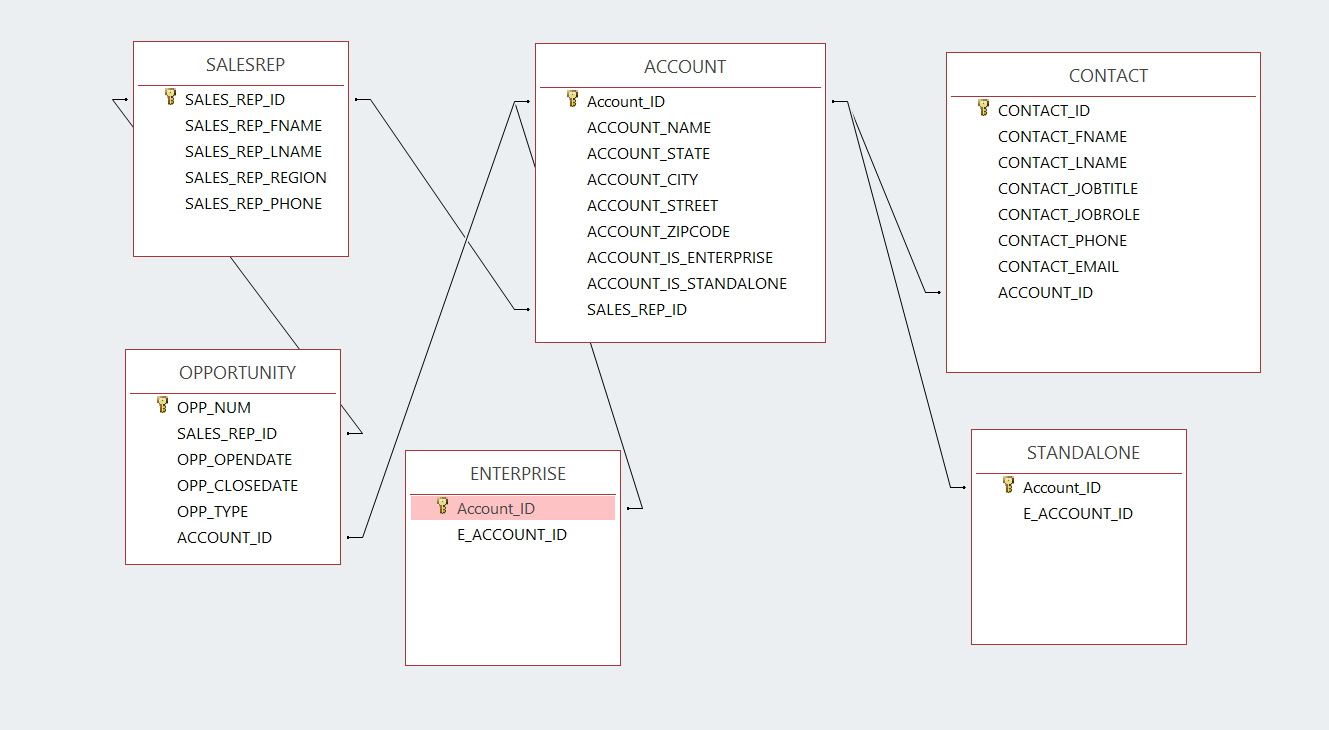
**Business Rules:**

* Every account is either a stand alone account or an enterprise account
* An enterprise account can have none or many stand alone accounts
* An account can have none or many contacts and each contact is only associated with one account
* A sales rep may or may not have an opportunity, however they could have many opportunities.
* An opportunity must be associated with one and only one sales rep
* An opportunity type can be a new sale, renewal, or expansion
* A sales rep can manage none or many accounts, an account can be managed by none or only one sales rep
* For every account there can only be one opportunity
* Contacts should include name, phone number, email, job title, and job role
* Accounts should include name, state, city, ctreet, zip code
* Information on Sales Representatives should include name, region, and phone number
* Information on Opportunity should include opportunity open date, opportunity close date

**ER Diagram**

****

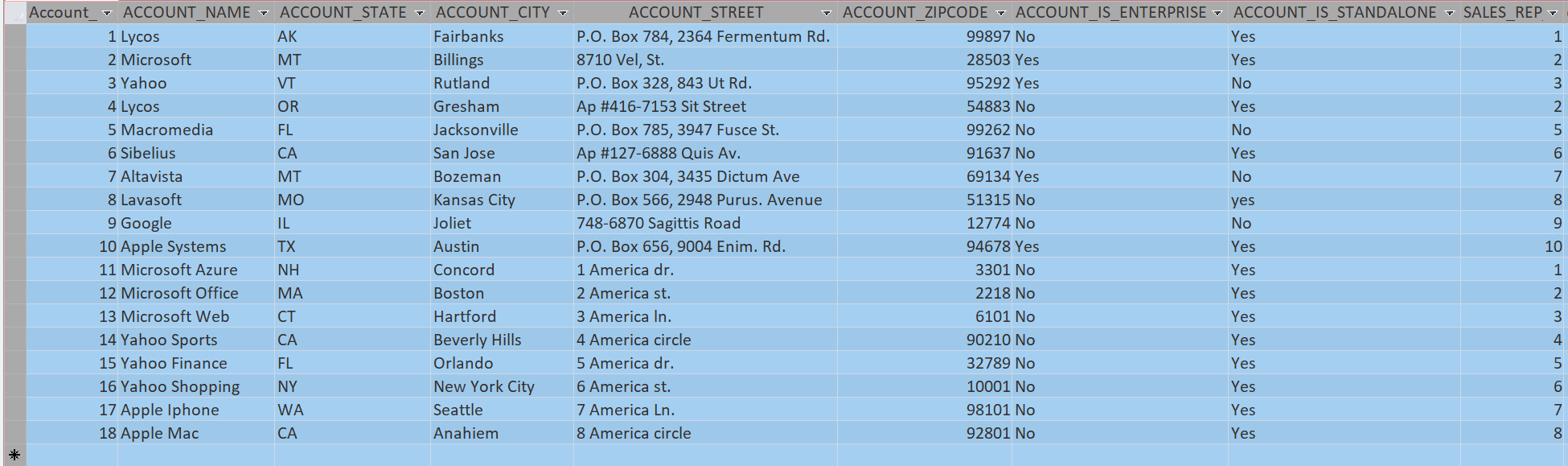
**Relational Schema**



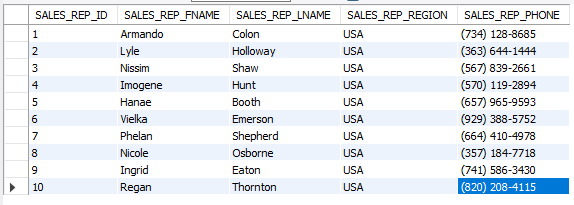
**Database Tables**

We created 6 tables in total, one for each entity( SALESREP for the Sales Representative entity, ACCOUNT for the ACCOUNT entity, OPPORTUNITY for the OPPORTUNITY entity, CONTACT for the CONTACT entity, ENTERPRISE for the ENTERPRISE ACCOUNT entity, and STANDALONE for the STANDALONE ACCOUNT entity. Screenshots of each table are displayed below.

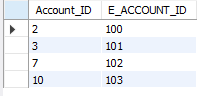
**ACCOUNT**

****

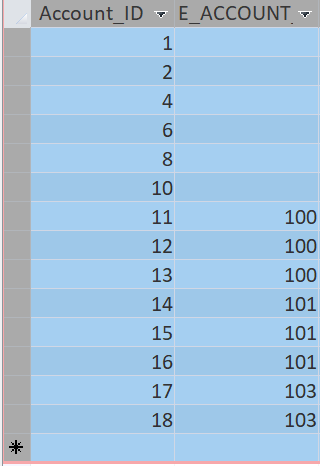
**SALESREP**

****

**ENTERPRISE**



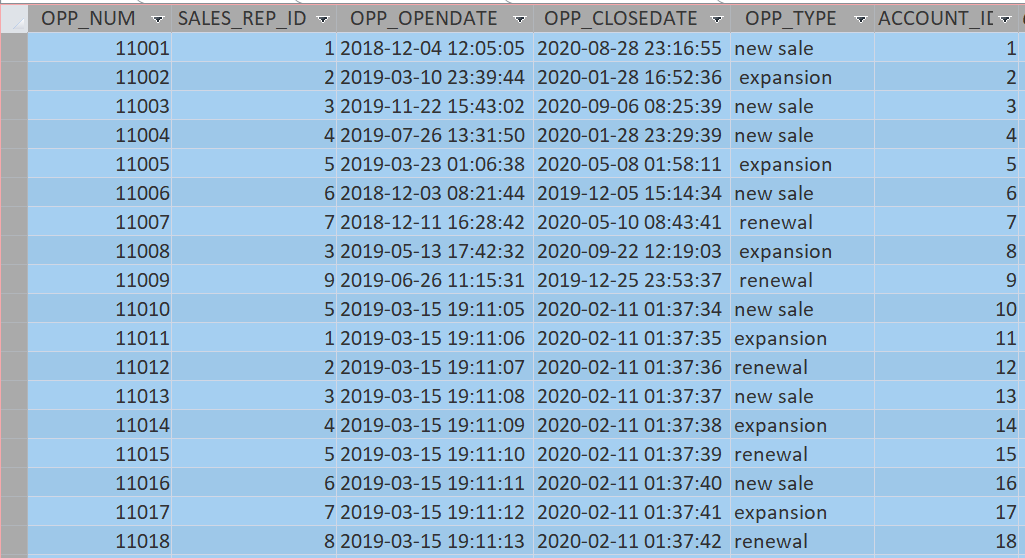
**STANDALONE**



**CONTACT**



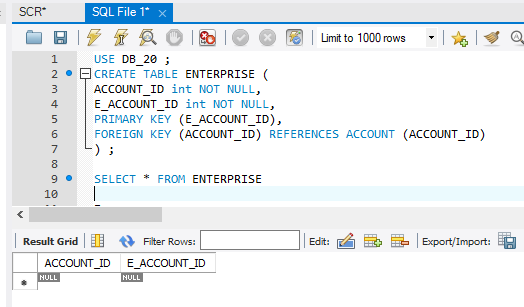
**OPPORTUNITY**

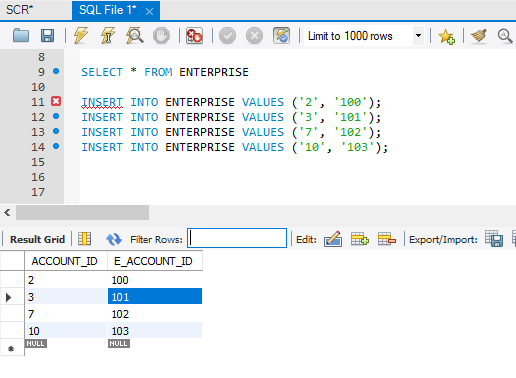
****

**Creating the Tables and Loading the Data**

We started creating our tables programmatically using the Create table function on sql.

Then we continued by identifying our keys ( Primary and Foriegn) and establishing relationships. Then we loaded our data using the INSERT function. We created some of our tables programmatically and created the rest automatically by uploading them. This was due to the size of the dataset and the time constraint we faced.

****

****

**Queries & Results**

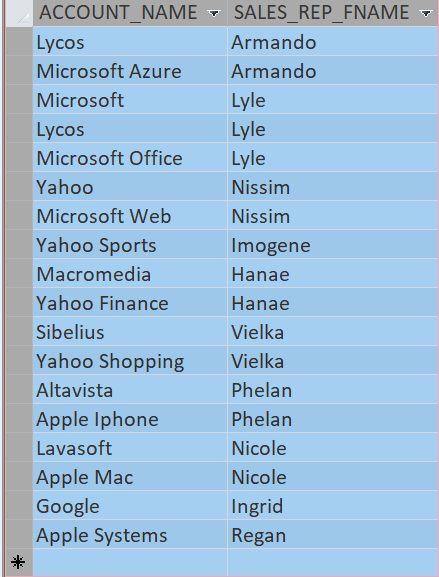
**Query 1: Display every account name and the sales rep belonging to the account.**

**SQL Code:**

SELECT ACCOUNT.ACCOUNT\_NAME, SALESREP.SALES\_REP\_FNAME

FROM SALESREP INNER JOIN ACCOUNT ON SALESREP.SALES\_REP\_ID = ACCOUNT.SALES\_REP\_ID;

**OUTPUT:**

****

**Query 2: Display every account name and contacts first name and job title within each account. Sort by account name in ascending order.**

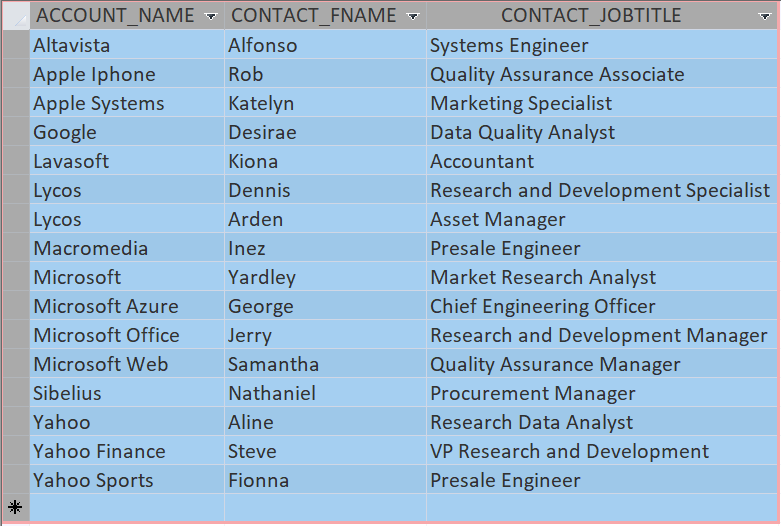
**SQL Code:**

SELECT ACCOUNT.ACCOUNT\_NAME, CONTACT.CONTACT\_FNAME, CONTACT.CONTACT\_JOBTITLE

FROM ACCOUNT INNER JOIN CONTACT ON ACCOUNT.Account\_ID = CONTACT.ACCOUNT\_ID

ORDER BY ACCOUNT.ACCOUNT\_NAME;

**OUTPUT:**

****

**Query 3: Display every account name, the sales rep first name belonging to the account, the opportunity number, opportunity open and close date, as well as the opportunity type. Sort by account name in ascending order.**

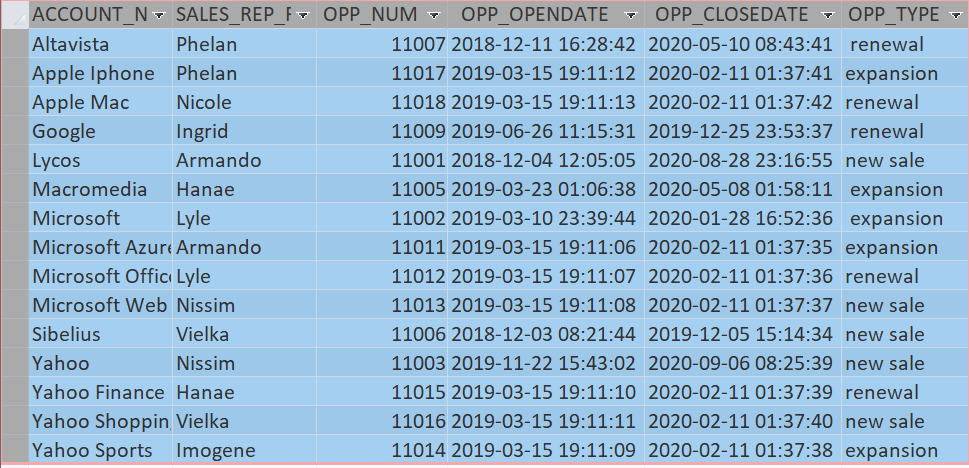
**SQL Code:**

SELECT ACCOUNT.ACCOUNT\_NAME, SALESREP.SALES\_REP\_FNAME, OPPORTUNITY.OPP\_NUM, OPPORTUNITY.OPP\_OPENDATE, OPPORTUNITY.OPP\_CLOSEDATE, OPPORTUNITY.OPP\_TYPE

FROM (SALESREP INNER JOIN ACCOUNT ON SALESREP.SALES\_REP\_ID = ACCOUNT.SALES\_REP\_ID) INNER JOIN OPPORTUNITY ON (SALESREP.SALES\_REP\_ID = OPPORTUNITY.SALES\_REP\_ID) AND (ACCOUNT.Account\_ID = OPPORTUNITY.ACCOUNT\_ID)

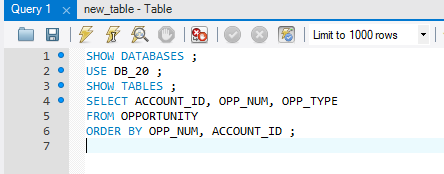
ORDER BY ACCOUNT.ACCOUNT\_NAME;

**OUTPUT:**

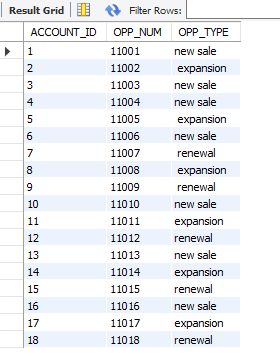
****

**Query 4: Display account id, opportunity number, and opportunity type. Sort by opportunity number then by account id in ascending order.**

**SQL Code:**

****

**OUTPUT:**

****

**Query 5: Display all of the stand alone account names under an enterprise account and the enterprise account name. Order by standalone account name in ascending order.**

**SQL Code:**

SELECT ACCOUNT.ACCOUNT\_NAME, [ENTERPRISE LIST].ACCOUNT\_NAME

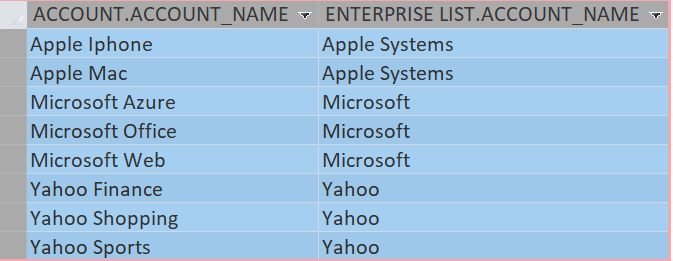
FROM [ENTERPRISE LIST], STANDALONE INNER JOIN ACCOUNT ON STANDALONE.Account\_ID = ACCOUNT.Account\_ID

WHERE (((STANDALONE.E\_ACCOUNT\_ID)=[ENTERPRISE LIST].[E\_ACCOUNT\_ID]))

ORDER BY ACCOUNT.ACCOUNT\_NAME;

Note: in this query we made another table called enterprise list that contained account name and e\_account\_id that corresponded to the account name

**OUTPUT:**

****

**Conclusion & Lessons Learned**

This project was a great learning experience for us. The initial purpose and objective proved to be a difficult one as Jillian said it might take some time to think about. In making dummy data we discovered that there was so much information and attributes we could have included in each table but for simplicity we decided to trim the data down. Another lesson learned was in trying to “roll up” the accounts. There were several ways to do this and each seemed to be a difficult and gruesome task. We came to a point where we were able to make another table consisting of the enterprise account information to help create a list of each account and their enterprise account. Overall this project was a success. With Jillian’s help we were able to find a solution to her problem and was satisfied with our work.